# Using a Serial ROM with the 21555 Non-Transparent PCI-to-PCI Bridge

## Application Note

***November 2002***

Order Number: 278383-002



**Application Note**

# Contents



## Figures



## Tables

# 1.0 Introduction

The serial ROM interface is used to preload data into the 21555 bridge configuration registers with vendor specific values. The serial ROM can also be used to support the Vital Product Data (VPD) interface as described in the *PCI Local Bus Specification, Revision 2.3*.

This application note describes the serial ROM hardware setup and explains the preload sequence of the serial ROM. Example code for programming the serial ROM through command and status register (CSR) access is provided. A description of the VPD operation is also provided.

# 2.0 Hardware Description

This section describes the supported serial ROMs and the hardware configuration.

## 2.1 Supported Serial ROMs

The serial ROM interface of the 21555 bridge supports byte-organized 4096-bit (512-byte) Microwire serial ROM. The recommended Microchip device is either the 93LC66 (configured in byte mode) or the 93LC66A. The clock input to the serial ROM is the primary PCI clock with a maximum frequency of 33 MHz, divided by 34 or 66 MHz divided by 68 if P_M66ENA is low. The duty cycle of the clock is approximately 50%.

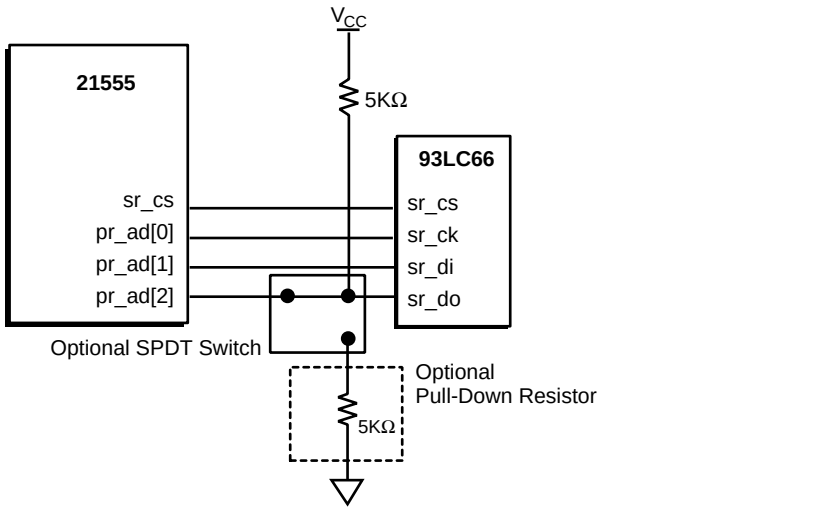
## 2.2 Hardware Configuration

The serial ROM interface consists of four signals as listed in Table 1. Chip select, **sr_cs**, is a dedicated signal. The remaining three signals are multiplexed with the parallel ROM control signals. Figure 1 shows the hardware set up for the serial ROM interface. The signal **pr_ad[2]** is the serial ROM output during accesses to the ROM. If a serial ROM is not present, **pr_ad[2]** should be tied low through an external resistor. When **pr_ad[2]** is tied to a serial ROM that is programmed with the correct preload enable sequence, the 21555 bridge will conduct a configuration register preload from the serial ROM. This takes approximately 570 ms. During the serial preload period, all configuration register accesses to the 21555 bridge receive a target retry.

**Table 1. Serial ROM Interface Signals**

| Name | Type | Description | 21555 Pin |
|---|---|---|---|
| **sr_cs** | Output | Serial ROM chip select | **sr_cs** |
| **sr_ck** | Output | Serial ROM clock | **pr_ad[0]** |
| **sr_di** | Output | Serial ROM data in | **pr_ad[1]** |
| **sr_do** | Input | Serial ROM data out | **pr_ad[2]** |

**Figure 1. Serial ROM Hardware Configuration**




# 3.0 Serial ROM Preload Operation

This section describes the preload sequence, the preload configuration registers, and the critical configuration registers.

## 3.1 Preload Sequence

After the 21555 bridge completes a chip reset and deasserts **s_rst_l**, it initiates a serial ROM read to perform a configuration register preload. The 21555 will only perform the preload operation if the first two bits read, corresponding to bits [7:6] of byte 0, are 10b. Otherwise the serial ROM read is terminated and all registers remain at their reset values. The preload sequence performs the following:

- Selects the size and type of downstream, upstream, and ROM address windows by preloading the address setup configuration registers.
- Loads read-only configuration values such as subsystem vendor ID, subsystem ID, class code, MAX_LAT, and MIN_GNT.
- Enables device specific features (for example, the read and write queue tuning thresholds, additional error modes, and $I_2O$ enable) in the chip control 0 and 1 configuration registers.
- Configures the bus master priority levels of the secondary bus arbiter.
- Enables or disables the PCI interface signal SERR# for specific error conditions.
- Sets up the power management register interface for the subsystem, including supported power states and PME# enable.
- Clears or sets the primary lockout bit in the chip control register, which enables or retries primary configuration accesses, respectively.

## 3.2 Preload Configuration Registers

The serial preload initializes those configuration registers not configured by standard plug and play initialization code. Table 2 provides a list of the preloadable registers. Not all of the bits in the sequence are actually used. Unused bits must be set to 0.

## Table 2. Serial Preload Format

| ROM Byte Offset | Register Offset | Description |
|---|---|---|
| 00h | — | ROM present (bits [7:6] = 10b to enable preload) |
| 03:01h | — | Reserved |
| 04h | 09h | Primary programming interface |
| 05h | 0Ah | Primary sub class code |
| 06h | 0Bh | Primary baseclass code |
| 08:07h | 2D:2Ch | Subsystem vendor ID register |
| 0A:09h | 2F:2Eh | Subsystem ID register |
| 0Bh | 3Eh | Primary minimum grant register |
| 0Ch | 3Fh | Primary maximum latency register |
| 0Dh | 49h | Secondary programming interface register |
| 0Eh | 4Ah | Secondary sub class code register |
| 0Fh | 4Bh | Secondary base class code register |
| 10h | 7Eh | Secondary minimum grant register |
| 11h | 7Fh | Secondary maximum latency register |
| 15:12h | 13:10h | Downstream memory 0 setup register |
| 19:16h | 1B:18h | Downstream I/O or memory 1 setup register |
| 1D:1Ah | 1F:1Ch | Downstream memory 2 setup register |
| 21:1Eh | 23:20h | Downstream memory 3 setup register |
| 25:22h | 27:24h | Downstream memory 3 setup register(upper 32 bits) |
| 27:26h | C3:C0h | Primary expansion ROM setup register |
| 2B:28h | 5B:58h | Upstream I/O or memory 0 setup register |
| 2F:2Ch | 5F:5Ch | Upstream memory 1 setup register |
| 31:30h | CD:CCh | Chip control 0 register |
| 33:32h | Cf:CEh | Chip control 1 register |
| 35:34h | D3:D2h | Arbiter control register |
| 36h | D4 | Primary SERR# disables register |
| 37h | D5 | Secondary SERR# disables register |
| 3F:38h | See[a] | PCI power management data register values |
| 40h | — | Reserved |
| 42:41h | See Table 3 | PCI power management and BiST |

a. These 8 registers are hidden but can be read in the power management data register (offset E3h). The data select register (offset E1h) determines the byte in the data register.

Serial ROM bytes 42:41h, which contain the PCI power management setup information and BiST control, have the format shown in Table 3.

**Table 3. Power Management and BiST Preload Format**

| ROM Byte Offset | ROM Bits | Description | Register Offset | Register Bits |
|---|---|---|---|---|
| 41h | 1:0 | Reserved | — | — |
| 41h | 2 | BiST supported | — | — |
| 41h | 3 | Power management data register enable | — | — |
| 41h | 5:4 | Power management control and status register | E1:E0h | 14:13 |
| 41h | 7:6 | Power management capabilities register | DF:DEh | 1:0 |
| 42h | 0 | Power management capabilities register | DF:DEh | 2 |
| 42h | 1 | Power management capabilities register | DF:DEh | 5 |
| 42h | 7:2 | Power management capabilities register | DF:DEh | 14:9 |

## 3.3 Critical Configuration Registers

After a serial ROM preload, the initialization sequence can either immediately step to the host configuration, or the local processor can configure the secondary side of the 21555 bridge before allowing access by the host. The critical configuration registers, which should be set in the serial ROM preload, depend on which initialization sequence is utilized. If the host requires access immediately after the preload, the critical configuration registers are set as follows (all other locations in the serial ROM preload can be loaded with 0's for this initialization sequence):

- ROM present. Bits 7:6 set to a 10b to enable serial ROM preload.
- Primary/secondary sub classcode. Sets the subsystem device class type.
- Subsystem ID and subystem vendor ID. Provides the vendor specific ID for subsystem.
- Upstream and downstream base address setup registers. Sets the address range size and type before host configuration.
- Primary expansion ROM setup register. Sets the address range size for expansion ROM.
- Chip control 0. Must clear the primary lockout bit (bit 10).

If the local processor configures the subsystem after a serial ROM preload and before the host gains access to the 21555 bridge, the following configuration registers should be set during preload (all other locations in the serial ROM preload can be loaded with 0's for this initialization sequence):

- ROM present. Bits 7:6 set to a 10b to enable serial ROM preload.
- Upstream base address setup registers. Sets the address range size and type before secondary processor configuration.
- Chip control 0. Must set the primary lockout bit to a one (bit 10).

# 4.0 Serial ROM Operation by CSR Access

The 21555 bridge allows serial ROM access through CSR control. A serial ROM operation consists of the following three phases:

- Command (3 bits)
- Address (9 bits)
- Data (8 or more bits)

For a serial ROM write operation (opcode 01), the 21555 bridge drives the data to the serial ROM on the **sr_do** line. For a read operation (opcode 10), the serial ROM drives the data to the 21555 bridge on the **sr_do** line.

In addition to these two operations, the 21555 bridge supports the following op codes:

- Write enable (opcode 00)
- Write disable (opcode 00)
- Write all (opcode 00)
- Erase all (opcode 00)
- Erase (opcode 11)

When writing the opcode and address to the ROM address register, bits [10:9] contain the opcode and bits [8:0] contain the address. For opcode 00, a byte address is not required and the two most significant address bits [8:7] decode the following commands:

- Write disable (00)
- Write all (01)
- Erase all (10)
- Write enable (11)

## 4.1 Serial ROM Write

To perform a serial ROM write operation, write enable the serial ROM because most ROM's typically power up in the ERASE/WRITE disable state. After performing this operation once, the serial ROM will remain enabled for programming unless a write disable command is issued or Vcc is removed from the serial ROM. To perform a write enable, the initiator should first make sure that the parallel and serial ROM start and busy bits are both clear in the ROM control CSR (offset 0CFh). Then perform the following steps:

1. Write byte address and opcode (000180h) to the ROM address register (0CCh).
2. Write 1 to the serial ROM start bit and write 0 to the parallel ROM start bit in the ROM control CSR during the same CSR access.
3. When the serial ROM start bit is read as a zero, the operation is complete.

Example code for the write enable operation is provided as follows:

```
/* setup 21555 for write enable, base = 21555 memory mapped CSR base address */
PUT32(base+0xCC,SROM_OpGeneral|SROM_GeneralWriteEnable);
/* kick the start bit */
PUT8(base+0xCF,RCR_SROMStartBusy);
/* wait for busy bit to deassert, ms delay so we are not too aggressive */
while ((value=GET8(base+0xCF)) & RCR_SROMStartBusy) delay(1);
```

After the serial ROM is enabled for programming, the write operation can begin. The steps required to write to the serial ROM are listed as follows:

1. Write the byte address and opcode (000200h, address 0, opcode 01) to the ROM address register (0CCh).
2. Write the 8-bit data (80h, enable serial preload) in the ROM data CSR (0CAh).
3. Write the serial ROM start bit to a 1 and the parallel ROM start bit to a 0 in the ROM control CSR during the same CSR access.
4. When the serial ROM start bit is read as a 0 and the SROM_POLL bit of the ROM control CSR is set to a 1, a polling operation is initiated to test for write completion by writing the serial ROM start bit to a 1.
5. The SROM_POLL bit indicates the status of the polling operation. If it is read as a 1, the serial ROM should be polled again. If it is read as a 0, the operation is complete.

Example code for the write operation is provided as follows:

```
/* setup for write command. i = srom offset, sBuffer[i] = data to write */
PUT32(base+0xCC,i|SROM_OpWrite);
PUT8(base+0xCA,sBuffer[i]);
/* kick the start bit */
PUT8(base+0xCF,RCR_SROMStartBusy);
/* wait for busy bit to deassert */
while (GET8(base+0xCF) & RCR_SROMStartBusy);
/* wait for SROM to recover */
delay(4);

/* we are now in write mode, kick the start bit again to program the data*/
PUT8(base+0xCF,RCR_SROMStartBusy);
while (GET8(base+0xCF) & RCR_SROMStartBusy);
/* keep issuing a start command to see if the SROM has accepted the data */
/* by checking the poll bit */
while(GET8(base+0xCF) & RCR_SROMPoll)
```

```
{
PUT8(base+0xCF,RCR_SROMStartBusy);
while (GET8(base+0xCF) & RCR_SROMStartBusy);
}
```

## 4.2 Serial ROM Read

To perform a serial ROM read, both the parallel and serial ROM start and busy bits should be clear in the ROM control CSR. Then perform the following steps for a serial ROM read access:

1. Write the byte address and op code (000400h, address 0, opcode 10) in the ROM address CSR (offset: 0CCh).
2. Write the serial ROM start bit to a 1 and the parallel ROM start bit to a 0 in the ROM control CSR during the same CSR access.
3. When the serial ROM start bit is read as a 0, the 8-bit data from the ROM data register (offset: 0CAh may be read).

Example code for the read operation is provided as follows:

```
/* issue a read command with correct address (i) */
PUT32(base+0xCC,i|SROM_OpRead);

/* kick the start bit */
PUT8(base+0xCF,RCR_SROMStartBusy);

/* wait for busy the bit to deassert with 1 ms delay */
while (GET8(base+0xCF) & RCR_SROMStartBusy) delay(1);
/* read data */
value=GET8(base+0xCA);
```

## 4.3 Example Data File for Programming Serial ROM

MKSROM.EXE is a DOS based utility that can be used to program a blank serial ROM behind a 21555 bridge from PCI. A data file is used with this program to load the appropriate information into the serial ROM. This data file can be altered using any standard text editor. Example code for the data file is provided as follows:

```
;
; this file will create an SROM image
; This loads the registers with their reset
; values - BARs disabled, No I2O, no special features
;
[
; preload enable (sets bit 7 for preload enable)
:0 80
:1 00
:2 00
```

```
:3 00
; Primary Class Code
:4 00
:5 80
:6 06
; Subvendor IDs
:7 46
:8 00
:9 11
:A 10
; Primary Min GNT, Max LAT
:B 00
:C 00
; Secondary Class Code
:D 00
:E 80
:F 06
; Secondary Min GNT, Max LAT
:10 00
:11 00
;Downstream Mem 0 - CSRs only (Set a 4K window size)
:12 00
:13 F0
:14 FF
:15 FF
; Downstream Mem 1 or I/O (Set 256 byte I/O window size)
:16 01
:17 FF
:18 FF
:19 FF
; Downstream Mem 2 (Set 8MB memory window size)
:1A 08
:1B 00
:1C 80
:1D FF
; Downstream Mem 3 (Set 8MB memory window size)
:1E 08
:1F 00
:20 80
:21 FF
; Downstream Mem 3 Upper 32
:22 00
:23 00
:24 00
:25 00
; Expansion ROM (Set 1MB expansion ROM size)
:26 01
:27 F0
; Upstream Mem 0 or I/O (Set 256 byte I/O window size)
:28 01
:29 FF
:2A FF
:2B FF
; Upstream Mem 1 (Set 8MB memory window size)
:2C 08
:2D 00
:2E 80
:2F FF
```

```
; Chip Control 0
:30 00
; clear lockout bit
:31 00
; Chip Control 1
:32 00
; LUT disable,i2o disable
:33 00
; Arbiter control (Sets internal 21555 secondary bus request to high priority ring)
:34 00
:35 02
; System error disable
:36 00
:37 00
; Power management
:38 00
:39 00
:3A 00
:3B 00
:3C 00
:3D 00
:3E 00
:3F 00
:40 00
:41 00
:42 00
]
```

# 5.0 Vital Product Data

The 21555 bridge provides VPD support through the serial ROM interface. The upper 3072-bits of the serial ROM is used to store VPD data. The first 1024-bits of this space is designated as read only and cannot be written from the VPD serial ROM register interface. The upper 2048-bits are read/write locations that can be accessed through the VPD serial ROM register interface.

## 5.1 Reading VPD Information

A read can occur to any location in VPD space. Valid VPD addresses are 17F:000h when accessing from the VPD address register (offset by 1024-bits from serial ROM address register). To read VPD information from the serial ROM, perform the following steps:

1. Write the VPD address and VPD flag bits of the VPD address register (offset E7:E6h). The lower 9 bits of this register are the address bits and bit 15 should be set to a 0 signifying a read command. The 21555 bridge adds the VPD base address, 080h, to the VPD byte address in order to obtain the serial ROM address. The VPD address has no alignment criteria and can start on any byte boundary.
2. The VPD flag (bit 15) is polled. When the 21555 bridge returns a 1, the read is complete.
3. Read the VPD information from the VPD data register (offset EB:E8). Byte 0 will contain the data referenced by the VPD address register and bytes 3:1 include the successive bytes.

The 21154, regardless of the address, always performs a four-byte read. Therefore, if the VPD byte address is one of the last 3 bytes in VPD space, the serial ROM address wraps. The remaining 1, 2 or 3 bytes contain invalid data and should be ignored.

## 5.2 Writing VPD Information

A write can only occur in the last 2048-bits (256 bytes) of VPD space. Valid VPD write addresses are 17F:080h. To write VPD information to the serial ROM, perform the following steps:

1. Write the VPD data to the VPD data register (offset EB:E8). Byte 0 contains the data to be written to the location referenced by the VPD byte address. The values in bytes 3 through 1 of the VPD data register are written to successive locations in VPD space.
2. Write the VPD address and VPD flag bits of the VPD address register (offset E7:E6h). The lower 9 bits of this register contain the address bits. Bit 15 should be set to a 1 signifying a write command. The 21555 bridge adds the VPD base address, 080h, to the VPD byte address to obtain the serial ROM address. The VPD address has no alignment criteria and can start on any byte boundary.
3. The VPD flag bit is polled. When the 21555 bridge returns a 0, the write is complete.

Writes attempted to a location in the lower 1024-bits of serial ROM space will not be performed and the 21555 bridge will clear the flag bit immediately. If the VPD byte address is one of the last three byte locations in VPD space, the 21555 bridge will only complete those writes to the end of VPD space.

The 21555 bridge tracks whether the serial ROM is enabled for writes. If the serial ROM is write disabled and a VPD write is attempted, the 21555 bridge will first automatically perform a write enable operation to the serial ROM.